AMENDMENTS TO THE SPECIFICATION

Please amend paragraph 0003 on pages 1-2, as follows:

[0003] A multipurpose network device such as, for example, a combination router and file server performs[[,]] Transmission Control Protocol/Internet Protocol (TCP/IP) routing functions, file server functions, management functions, and other activities based on incoming packets. These various activities must compete for shared resources within the device. As noted above, among the scarce shared resources are CPU time. While traditional systems allocate CPU time by task prioritization, task prioritization does not consistently provide desired and predictable performance.

Please amend paragraph 0012 on page 5, as follows:

[0012] As used herein, a "flow" is deemed to be an aggregation of packets of a particular type or category regardless of source. Thus, if two different clients were sending packets directed to internet 122[[.]], the ~~The~~ packets from both clients would constitute one passthrough flow. It is desirable that all flows have some guaranteed Central Processing Unit (CPU) time to avoid starvation of the associated activity. Different types of packets will have different costs in terms of processor time required to service them. The packet load on the system depends on a number of factors. As noted above, one major distinction is between pass through flows (packets received by the network element 110) at one interface and passed through to another interface of network element 110 and flows to be handled internal to network element 110. Packets to be handled internally can further be differentiated between i) control and management packets used by a remote operator who monitors the system; and ii) packets to a file server within network

element 110. Accordingly, a non-exclusive list of packets expected in one embodiment includes: i) packets routed through the network element without encryption; ii) packets routed through the network element with encryption; iii) management and control packets; and iv) packets addressed to the file server.

Please amend paragraph 0013 on pages 5-6, as follows:

[0013] In one embodiment, the cost is used as a scaling factor to ~~normalized~~ normalize the load of a flow or the processor 114. The system load is given by the equation $\vec{L} = (L_1, L_2, \ldots L_N)$ where there are N flows, and N is an arbitrarily large number. The load in packets per second (pps) for each flow F_i is given by the equation $L_i = C_i \times I_i$ where I_i isthe input rate in pps and C_i is the cost scaling factor. Thus, L_i is express in normalized pps.

Please amend paragraph 0015 on pages 6-7, as follows:

[0015] Figure 1b is a bar diagram reflecting an example of load balancing in one embodiment of the invention. In this diagram, four flows are shown. Flow 1 has an allocated maximum steady state of ten packets, flow 2 has five packets, flow 3 has ~~three~~ eight packets and flow 4 has ~~ten~~ two packets. Thus, P for this system is twenty-five. The load for flow 1 is six packets, for flow 2 is five packets, for flow 3 is fourteen packets and for flow 4 is six packets. Thus, the aggregate load L is thirty-one. This reflects an overloaded condition. Looking to the individual flows, flow 1 has an unused capacity of 4 packets, flow 3 has excessive usage of 6 packets and flow 4 has an over usage of 4 packets. The appropriate scaling factor calculated using the equation set forth above is $[(8 + 2) + 4]/(14 + 6) =$

0.7. This indicates that flow 3 should be scaled down to ten packets and flow 4 should be scaled down to four packets. The numbers are arrived at using the scaling factor and an integer function, e.g., int (14 x 0.7) = int (9.8) = 10. Alternatively, a floor function could be used to make absolutely certain that the scaled load does not result in an overloaded condition. For example, floor (14 x 0.7) = floor (9.8) = 9. The integer function rounds to nearest integer while the floor function rounds to a next smaller integer.

Please amend paragraph 0018 on page 8, as follows:

[0018] At decision block 206, a determination is made if the aggregate load of all of the flows and exceeds a predicted steady state threshold. At decision block 208, if the aggregate flow does not exceed the steady state threshold, the determination is made if the processor is over utilized. If the processor is not over utilized, the system advances to the next time slice and no drop policy is employed at functional block 210. In one embodiment, a time slice is 200ms, other ~~embodiment~~ embodiments may employ longer or shorter time slices. If the load is greater than the steady state threshold at decision block 206, at decision block 214 a determination is made if the processor is under utilized. If the processor is under utilized the steady state threshold is raised to more efficiently use the processor. In one embodiment, there is a range in which the steady state threshold may be established. In one such embodiment, the rate at which the steady state threshold is reduced responsive to over utilization exceeds the rate at which the steady state threshold is increased responsive to under utilization. In another embodiment, the steady state threshold is

fixed and unchangeable after manufacture. In such an embodiment, blocks 212-216 are effectively omitted.

Please amend paragraph 0019 on page 9, as follows:

[0019] After the steady state threshold is lowered or if at decision block 214 the processor is not under utilized, or after the steady state threshold is raised, a drop policy for the next time slice is calculated at functional block ~~216~~ 218. Then at functional block 220, the processor selects an appropriate drop buffer reflecting the drop factor calculated at functional block 218. At functional block 222, the system advances at the next time slice and applies the drop policy reflected in the previously selected drop buffer. Accordingly, the drop policy for a time slice T_1 is established based on actual traffic in time slice T_0.